UpLink - A Linked Data Editor for RDB-to-RDF Data

Hert, Matthias ; Marsella, Sergio ; Reif, Gerald ; Gall, Harald C

Abstract: Linked Data builds a machine-processable Web of Data based on a large and growing number of RDF datasets and typed links among them. For the human user, Web-based interfaces were developed to enable browsing and editing Linked Data that is stored as native RDF. However, the majority of data on the current Web is stored in Relational Databases (RDB). This is a challenge for Linked Data browsers and especially for Linked Data editors. In this paper, we present UpLink which is to the best of our knowledge the first Linked Data editor for RDB-to-RDF data, i.e., RDF data that is mapped on demand from a RDB. We further present usage scenarios to demonstrate that UpLink supports the basic CRUD operations for editing Linked Data.

DOI: https://doi.org/10.1145/2063518.2063539

UpLink – A Linked Data Editor for RDB-to-RDF Data

Matthias Hert
Department of Informatics
University of Zurich
hert@ifi.uzh.ch

Sergio Marsella
Department of Informatics
University of Zurich
sergio.marsella@uzh.ch

Gerald Reif
innovation process technology
gerald.reif@ipt.ch

Harald C. Gall
Department of Informatics
University of Zurich
gall@ifi.uzh.ch

ABSTRACT

Linked Data builds a machine-processable Web of Data based on a large and growing number of RDF datasets and typed links among them. For the human user, Web-based interfaces were developed to enable browsing and editing Linked Data that is stored as native RDF. However, the majority of data on the current Web is stored in Relational Databases (RDB). This is a challenge for Linked Data browsers and especially for Linked Data editors. In this paper, we present UpLink which is to the best of our knowledge the first Linked Data editor for RDB-to-RDF data, *i.e.,* RDF data that is mapped on demand from a RDB. We further present usage scenarios to demonstrate that UpLink supports the basic CRUD operations for editing Linked Data.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Graphical User Interfaces*

General Terms

Design

Keywords

Linked Data editor, RDB-to-RDF mapping, OntoAccess

1. INTRODUCTION

The Linked Open Data project[1] is one of the most prominent success stories of applying Semantic Web technologies. A large and growing number of datasets is made available on the Web of Data following the Linked Data principles [2]. The primary goal of Linked Data is to enable a global data space [5] that is machine-processable, *i.e.,* individual datasets and the typed links between them can be processed without human intervention. However, interfaces for the human user were identified as important for the widespread adoption as they provide the immediate gratification for information providers of seeing the results of their efforts [3]. These so-called Linked Data browsers enable the human user to navigate through datasets by following links expressed as RDF triples. Over the years, these browsers evolved into Linked Data editors that in addition to read-only browsing of the data also provide editing capabilities. However, the state-of-the-art browsers and editors for Linked Data require that the data is available as native RDF (*e.g.,* in a RDF triple store).

The majority of data on the current Web is stored in Relational Databases (RDB) [7]. Therefore, bridging the conceptual gap between the relational model and RDF is required to make the data available on the Semantic Web. RDB-to-RDF mapping approaches were developed that include Linked Data views on the relational data, but to the best of our knowledge none provides editing capabilities.

The contribution of this paper is the Linked Data editor UpLink that operates on RDB-to-RDF data, *i.e.,* RDF data mapped on demand from a RDB. It builds on OntoAccess [9], a RDB-to-RDF mediation platform that provides RDF-based read and write access to the relational data.

The remainder of this paper is structured as follows. In Section 2, we briefly introduce the OntoAccess platform for RDF-based read and write access to RDBs as well as its bidirectional RDB-to-RDF mapping language R3M. Section 3 describes the implementation of our Linked Data editor UpLink while Section 4 presents usage scenarios to demonstrate that UpLink supports the basic CRUD operations for editing Linked Data. Related work in Linked Data browsers and editors is reviewed in Section 5. Section 6 summarizes the paper and provides an outlook on future work.

2. ONTOACCESS

OntoAccess [9] is a bidirectional RDB-to-RDF mediation platform, *i.e.,* it enables RDF-based read and *write* access to RDBs. It employs a mediation approach to provide data access via on demand request translation. This results in a cooperative use of the data in RDF-based as well as relational applications. In addition, mediation allows one to further exploit the advantages of the well established database technology such as query performance, scalability,

[1] `http://linkeddata.org/`

I-SEMANTICS 2011, 7th Int. Conf. on Semantic Systems, Sept. 7-9, 2011, Graz, Austria

transaction support, and security.

Implemented as an extensible platform, OntoAccess encapsulates the RDB-to-RDF translation logic in the core layer which provides the foundation of an extensible set of data access interfaces in the interface layer. This enables support for various data access approaches, *e.g.,* to translate SPARQL/Update requests as described in [10].

RDB-to-RDF mappings in OntoAccess are encoded in R3M [10], a bidirectional mapping language developed specially for OntoAccess that incorporates the requirements of RDF-based write access to RDBs [11]. It contains the mappings of tables to classes and attributes to properties as well as information about integrity constraints.

3. UPLINK

The main goal of UpLink [12] is to expose the functionality of the OntoAccess platform in a modern Web interface for the human user. This paper focuses on the Linked Data support in UpLink. In this section, we present the requirements and the implementation of the UpLink prototype.

The non-functional requirements for UpLink were to develop a modern Web application that provides an intuitive and easy to use interface for human users. It should incorporate state-of-the-art concepts in Web development such as asynchronous communication with the server and client-side processing via JavaScript. The functional requirements of UpLink were to expose the functionality of the OntoAccess platform in a Web interface. The prototype should implement support for browsing and editing Linked Data as well as for the SPARQL and SPARQL/Update languages. These features should be exposed in the Web interface for human users and via a HTTP endpoint for software agents. Furthermore, UpLink should be extensible to support additional data access interfaces provided by OntoAccess.

The implementation of UpLink is based on Grails,[2] a modern Web application framework that fosters agile development methodologies. It combines proven technologies with sensible defaults in a style called *convention over configuration* to simplify common development tasks. Although the UpLink prototype is extensible to support various data access interfaces, we limit the discussion in this paper to the interface for browsing and editing Linked Data.

Figure 1 depicts the UpLink Linked Data browser and editor. UpLink provides means to navigate a dataset from the vocabulary concepts down to the triples associated with an individual instance.

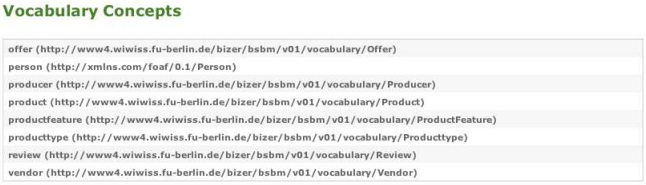

Figure 2: Concepts View

Figure 2 shows the *concepts view* that lists all vocabulary concepts used in the RDB-to-RDF mapping. Each concept URI is a link that opens the *instances view* for that concept as depicted in Figure 3.

[2] http://grails.org/

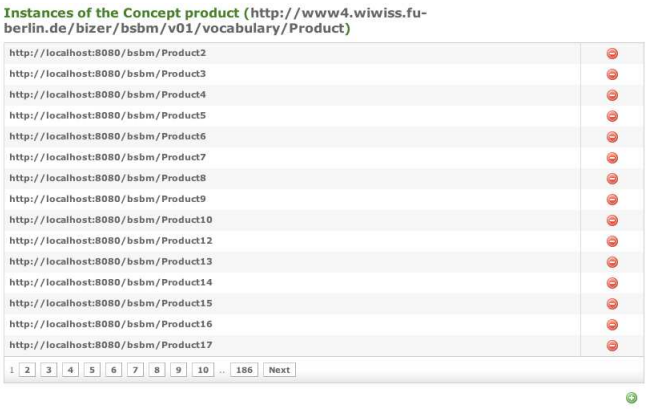

Figure 3: Instances View

In this view, all instances of the current concept are listed. As there can be a large number of instances, pagination is used to retain clarity. The *instances view* further provides editing features. The delete icon (⊖) displayed after each instance URI deletes all triples of this resource, *i.e.,* all triples that have this resource as the subject. The add icon (⊕) displayed at the bottom of the list of instances adds a new instance of the current concept. Clicking on that icon adds a new row to the instances list that requests the user to enter the URI of the new instance. After this instance URI is confirmed, the *resource view* is opened where triples can be added that belong to this resource, *i.e.,* that have the resource as the subject.

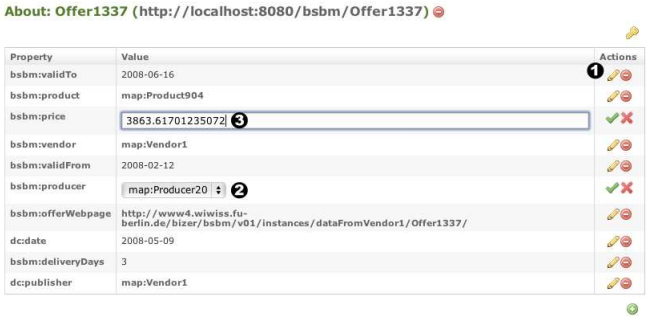

Figure 4: Resource View

Figure 4 depicts the *resource view* of an existing instance. This view can either be reached via the links in the *instances view* or directly via the URI of an instance. At the top of the page, the URI and a short name of the current resource are shown. This resource is the subject of all triples presented in this view. Below is a table with three columns. The first column contains the property of the triple represented in this row and the second column the object value which is either a literal or a URI. The third column contains the action icons (❶) for each triple. The edit icon (✎) switches the respective triple into *edit mode* where it is possible to edit the object value. If the object is an instance URI of a concept known from the RDB-to-RDF mapping, a drop-down list is rendered that contains all instances of that concept (❷).

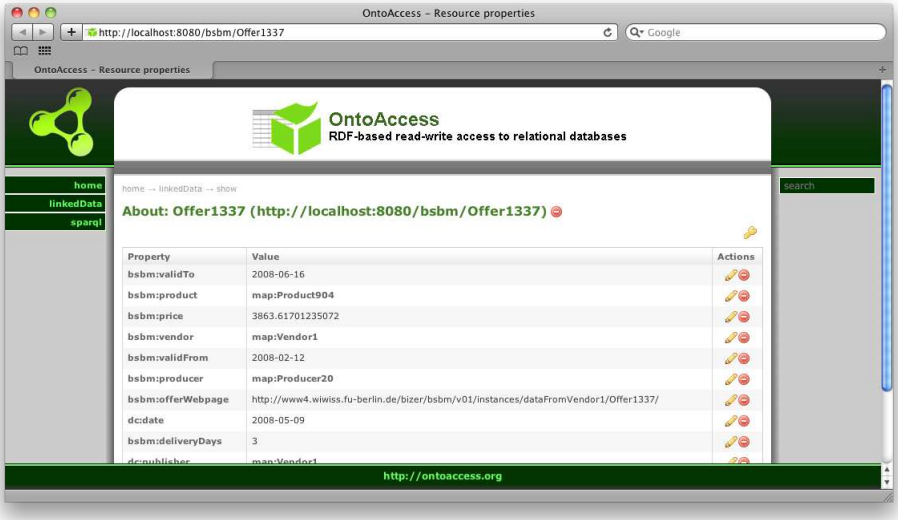

Figure 1: UpLink

Else, if the object is a literal or a URI not corresponding to a concept in the mapping, a simple editable text field (❸) is rendered. In *edit mode*, the action icons are replaced with icons for saving (✔) or canceling (✘) the edit operation. Both icons, if clicked, switch the triple back into the standard *view mode*. Note, that all these changes in the user interface happen without the need for a full page reload as they are implemented based on asynchronous communication with the server and client-side processing via JavaScript. The other icon in the actions column is the delete icon (⛔) which deletes the triple represented by the respective table row. Below the triples table resides the add icon (🟢) that adds a new triple with this resource as the subject. Clicking the icon adds a new row to the table and switches into *add mode*. In *add mode*, the property column contains a drop-down list that is filled with all possible properties for this resource. Matching properties are extracted from the RDB-to-RDF mapping of the current concept. After one of the properties is selected, the object column is rendered as described before in *edit mode*. The actions column also contains the save and cancel icons known from *edit mode*. Next to the URI of the current resource at the top of the page is another delete icon that performs the same action as the delete icon in the *instances view* – it deletes all triples that have this resource as the subject.

Changes in *edit mode* are submitted to OntoAccess on a triple-by-triple basis, *i.e.,* each changed triple generates one request. UpLink supports a *transaction mode* to send multiple changes in a single request. It is enabled by clicking the key icon (🔑) located above the triples table. In *transaction mode*, the key icon is replaced with the known save and cancel icons to commit or abort the transaction. Changes made in *transaction mode* are collected by UpLink on the client-side and are send to OntoAccess in a single request. OntoAccess treats this request also in a transactional manner, *i.e.,* it either applies all the changes or none.

The RDB-to-RDF mapping expressed in R3M is an essential resource in UpLink. The information in the mapping is exploited in many cases to improve the usability of the Web application. The *concept view* is generated from the mapping to restrict the list of concepts to those that can actually be mapped to a table in the RDB. In the *resource view*, the mapping is used to generate the properties that can be used in *add mode*, *i.e.,* that can be mapped to an attribute in the RDB. The mapping is further used in *edit mode* to decide on how to render the input controls of the object value, *i.e.,* as a drop-down list or a text field.

4. USAGE SCENARIOS

We present usage scenarios to demonstrate the capabilities UpLink provides for browsing and editing Linked Data. We use the synthetic dataset from the Berlin SPARQL Benchmark[3] (BSBM) which is built around an e-commerce use case. The scenarios are inspired by the BSBM *Explore and Update Use Case* [6]. There exist three scenarios.

Scenario 1: Correcting Product Data The first scenario is about correcting and extending the data of a product the user is interested in. First, the user looks for matching products with the SPARQL query Q1 of the BSBM suite (see [6] for details). From the query results a product is selected and opened in the *resource view*. While browsing the data of the product the user realizes that some triples are incorrect and that some are missing. Using the action icons in UpLink the user is able to edit the incorrect triples and by using the add icon the missing triples can be added.

Scenario 2: Adding a Product Review The second scenario is about adding a review of an existing product. The user first opens the *concepts view* and clicks on the concept `review` which opens the *instances view* of that concept. Here, the add icon is used to add

[3] `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/`

a new instance of the `review` concept. After entering the URI of the new instance the *resource view* is opened in *transaction mode*. There, the user can add the triples of the review instance and submit them to OntoAccess by saving the transaction.

Scenario 3: Deleting an Offer The third scenario is about deleting an offer that is no longer valid. First, the user looks for matching offers with the SPARQL query Q10 of the BSBM suite (see [6] for details). From the query results the user selects a product and opens it in the *resource view*. After checking that it is the intended offer, the user uses the delete icon to delete the resource and all of its triples.

These usage scenarios demonstrate as a proof of concept that UpLink supports the basic CRUD[4] operations on RDB-to-RDF data, *i.e.,* it allows to browse, edit, add, and delete Linked Data mapped on demand from a RDB.

5. RELATED WORK

Linked Data interfaces for the human user were identified as important for the widespread adoption as they provide the immediate gratification for information providers of seeing the results of their efforts [3]. In this section, we present Linked Data browsers and editors for native RDF data as well as Linked Data browsers for RDB-to-RDF data.

Interfaces for Native RDF Data The Disco Hyperdata Browser[5] is a simple Linked Data browser for navigating the Web of Data as an unbound set of data sources. A table of property-value pairs is displayed together with an indication of the source of the data. Resource URIs are rendered as hyperlinks to enable browsing through the Web of Data.

VisiNav[6] [8] is a system to search and navigate large amounts of Web data. Users can incrementally assemble complex queries from a set of atomic operations. The results can be visualized in detail, list, table, timeline, and map views.

Tabulator Redux [3] is a Linked Data editor that allows modifications and additions of RDF data within the browsing interface. Changes are relayed to the server on a triple-by-triple basis. One of the contributions of Tabulator Redux is a HTTP and SPARQL/Update-based protocol between an editor and incrementally editable resources stored in a 'data wiki'.

Interfaces for RDB-to-RDF Data D2R Server [4] is an approach for publishing existing RDBs on the Semantic Web. It enables the read-only browsing of relational data as Linked Data and also supports the SPARQL query language. Triples belonging to a resource are rendered as property-value pairs in a table on the resource's page.

Triplify [1] is a light-weight approach to publish Linked Data from RDBs. It is based on mapping HTTP-URI requests onto RDB queries and translating the resulting relations into RDF statements.

[4]Create, Read, Update, Delete

[5]`http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`

[6]`http://visinav.deri.org/`

6. CONCLUSION

In this paper, we presented the Linked Data editor UpLink. Based on the RDB-to-RDF mediation platform OntoAccess, it enables browsing and editing Linked Data mapped on demand from a RDB. We presented the prototype implementation and showed as a proof of concept that the basic CRUD operations for editing Linked Data are supported. We described how certain scenarios such as correcting existing data, adding new data, and deleting existing data can be performed in UpLink.

Future work is concerned with investigating federation aspects in UpLink. While the current prototype is limited to a single RDB as the data source, we plan to extend UpLink to multiple RDB-based and native RDF data sources, incorporating aspects of query and update federation.

7. ACKNOWLEDGMENTS

Partial support provided by Swiss National Science Foundation award number PDAMP2-122957.

8. REFERENCES

[1] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify – Light-Weight Linked Data Publication from Relational Databases. In *Proc. of the Int'l World Wide Web Conf.*, 2009.

[2] T. Berners-Lee. Linked Data. `http://www.w3.org/DesignIssues/LinkedData.html`, 2009. Last visited July 2011.

[3] T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, E. Prud'hommeaux, and M. C. Schraefel. Tabulator Redux: Browsing and Writing Linked Data. In *Proc. of the Workshop on Linked Data on the Web*, 2008.

[4] C. Bizer and R. Cyganiak. D2R Server – Publishing Releational Databases on the Semantic Web. In *Proc. of the Int'l Semantic Web Conf.*, 2006.

[5] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *Int'l Journal on Semantic Web and Information Systems*, 2009.

[6] C. Bizer and A. Schultz. Berlin SPARQL Benchmark (BSBM) Specification – V3.0. `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/`, 2010. Last visited July 2011.

[7] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured Databases on the Web: Observations and Implications. *SIGMOD Record*, 2004.

[8] A. Harth. VisiNav: A System for Visual Search and Navigation on Web Data. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 2010.

[9] M. Hert. Relational Databases as Semantic Web Endpoints. In *Proc. of the European Semantic Web Conf.*, 2009.

[10] M. Hert, G. Reif, and H. C. Gall. Updating Relational Data via SPARQL/Update. In *EDBT Workshop Proc.*, 2010.

[11] M. Hert, G. Reif, and H. C. Gall. A Comparison of RDB-to-RDF Mapping Languages. In *Proc. of the Int'l Conf. on Semantic Systems*, 2011.

[12] S. Marsella. Uplink – A Linked Data Interface for OntoAccess. Master's thesis, University of Zurich, 2011.